Duplication Scheduling of Periodic Tasks Based on Precedence Constraints and Commnunication Costs in Distributed Real-Time Systems

Mi-Kyoung Park† and Chang-Soo Kim††

ABSTRACT

Parallel tasks in distributed real-time systems can be divided into several subtasks and be executed in parallel according to their real-time attributes. But, it is difficult to gain the optimal solution which is to allocate a tasks deadline into the subtasks deadline while minimizing the subtasks deadline miss. In this paper, we propose the algorithm that allocates deadlines into each subtask, according to the attributes of each subtask(i.e. using communication time and execution time to periodic tasks). Also, we suggest a processor mapping algorithm that considers the communication time among the processors and the effective duplication algorithm which is allocated to the identical processor for the purpose of improving the communication time between the subtasks. We can obtain a result that reduces IPC (Inter-Processor Communication) time and uses the idle processor through applying effective real-time attributes to FUTD(Fully connected, Unbounded Task Duplication) algorithms. As a result, we can improve the average processor utilization.

분산 실시간 시스템에서 우선순위와 통신비용을 고려한 주기적 타스크들의 중복 스케쥴링

박미경† · 김창수††

요 약

분산 실시간 시스템에서 타스크들은 여러개의 서브 타스크들로 분할되어지고 그들의 실시간 특성들에 따라 병렬로 실행되지만, 이러한 서브 타스크들의 마감시간 분실을 최소화하면서 타스크 마감시간을 서브 타스크에 할당하는 최적의 해를 얻기란 어렵다. 본 논문에서는 주기적 타스크들의 통신시간과 수행시간을 이용해서 각 서브 타스크들의 속성에 따라 마감시간을 할당하는 알고리즘을 제시한다. 또한, 처리기들간의 통신시간을 고려한 처리기 사상 알고리즘과 서브 타스크들간의 통신시간을 개선하기 위해 동일한 처리기에 할당하는 효율적인 중복 알고리즘을 제시한다. 결과적으로 FUTD(Fully connected, Unbounded Task Duplication) 알고리즘에 효율적인 실시간 특성을 적용함으로써 IPC(Inter-Processor Communication) 시간을 줄이고 유휴 처리기를 이용해서 평균 처리기 이용률을 개선하였다.

1. Introduction

Recently, due to the development of computer networking technologies in various fields of study, a lot of research has focused on the distribution system. The resources or the data in other nodes can be used for the distribution systems, and the performance and the reliability of the system can be improved by using a large number of processors [16].

The fundamental characteristics that real-time systems offer are not only the logical correctness

† 부경대학교 전자계산학과 박사과정
†† 부경대학교 컴퓨터멀티미디어공학부 부교수

but also the exact results in a given time. The basic requirements real-time systems should show are not only a satisfying improvement of the system operation, but also how the tasks and the resources are effeectively scheduled in the proposed system [14].

In this paper, in consideration of the communication time and the execution time of each subtask, the deadlines are assigned according to the attributes of the subtasks, and we propose the processor mapping algorithm without the duplication of subtasks and the effective duplication algorithm based on the algorithms[3,18].

The rest of this paper is organized as follows. In section 2, related research will be discussed. In Section 3, the task and system model will be assumed. In Section 4, the deadline assignment algorithm of the subtask, whose periodic characteristics are taken into consideration, will be suggested. In Section 5, the duplication allocation algorithm and the non-duplication allocation algorithm will be proposed. In Section 6, the paper will be concluded.

2. Related Work

Generally, a task is divided into several subtasks for parallel processing in distributed and parallel real-time systems. The existing researches are only considered to the arrival time and the execution time for the parameters related to the deadlines [1,2,4]. There are the effective DIV-x and EQF methods for minimizing the subtasks deadline miss on the parallel and continuous execution [1,2]. Main problem of these methods is no considering the total states to the execution requirements of the subtasks and the communication time between them. Also, they do not consider the complicated states of the subtasks executed in parallel.

In the duplicate scheduling method, in the case where several nodes are linked like in the distributed system, a task can be allocated repeatedly to several processors to reduce the communication time of the tasks which require a mutual cooperation operation[19]. And let us refer to the duplication algorithms in the recent list scheduling field. The SDBS(Search and Duplication Based Scheduling) algorithm is proposed because it can make a task schedule with a processor through a method which can complete the execution in the shortest time in terms of the variable communication time[15]. The DFRN(Duplication First and Reduction Next) method is suggested, which brings about the decrease of the total execution time and the increase of the processor utilization rate in terms of the precedence relation and the task communication time[3]. The FUTD(Fully connected and Unbounded Task Duplication) method is suggested, which improves the time-complexity of the DFRN method [18]. In this paper we will be applied to the duplication method for the real-time task, because they have the merit of being the easiest to implement[3,18].

In this paper, the EDFRN(Effective Duplication First and Reduction Next), the duplication algorithm of the effective real-time task will be suggested, because it is expected to improve processor utilization and reduce the inter-processor communication time and task completion time.

3. A Task Model and System Environment

In this Section, we suggest the task model where more than two tasks with different periods can be executed in parallel, and will define the distributed system environment briefly.

3.1 A Task model

Generally, work in real-time systems consists of periodic and non-periodic tasks. Periodic tasks are the basic tasks that are executed at regular intervals. On the other hand non-periodic tasks are the transient tasks that come optionally.

Because most tasks in the real-time system are executed regularly, we will not include the non-periodic tasks in this paper. The attributes offered in the real-time systems are known basically before the execution [13]. The most important parameters in the periodic tasks are period, computation time and deadline etc[5,6,8,9]. Unlike the tasks with the soft deadlines, if those with the hard deadlines cannot be completed within a specified period of time, the entire system will fail. So they have to be completed in time.

The topology of distributed system is fully con nected into number of m nodes. Let the set of processors be $P=\{P_m|P_1, P_2, \cdots, P_m\}$ and the n tasks which are executed with m processors be $T=\{T_n|T_1, T_2, \cdots, T_n\}$. Then, $T_i=\{\tau_k|\tau_i, \tau_{i+1}, \cdots, \tau_k\}$ can consists of k subtasks. The notation $\tau_i=[\tau_i, \tau_2, \cdots, \tau_p]$ stands for number of p subtasks which can be execute in serial. And the subtasks $\tau_i(i>1)$ can not start to be executed before τ_{i-1}. Also, the notation $\tau_i=[\tau_1 \parallel \tau_2 \parallel \cdots \parallel \tau_q]$ represents the number of q subtasks which can be executed in parallel. If all subtasks of T_i complete the execution, then the task T_i execution will be considered to be completed. Formally, we define the classes of serial-parallel subtasks by the following rules:

ST_i : A single subtask which a predecessor and a successor do not exist in a periodic task is called a simple subtask.

SST_i : $SST_i=\{\tau_i|\tau_i, \tau_{i+1}, \cdots, \tau_k\}$, $\tau_i(1\le i\le k)$ is called serial subtasks.

PST_i : $PST_i=\{\tau_1 \parallel \tau_2 \parallel \cdots \parallel \tau_q\}$, $\tau_i(1\le i\le q)$ is called parallel subtasks.

The task T_i of this paper has the following attributes:

$ar(T_i)$ = the arrival time of T_i

$dl(T_i)$ = the deadline of T_i

$P(T_i)$ = the period of T_i

$ex(T_i)$ = the execution time of T_i

$comm(\tau_{i-1}, \tau_i)$ = the communication time between τ_{i-1} and τ_i.

In this paper, three periodic tasks with each different unique periods are extended to various subtasks in range of the least common multiple (LCM) of task periods to construct DAG(Direct Acyclic Graph) using serial-parallel subtasks [5,7,9,10]. When periodic tasks are executed by execution time and communication time as the input, the task model can be presented in Figure 1. To be structure, a subtask is represented as a circle, the character in the center portion of the circle represents the subtask identifier(id), the number in the right portion of the circle represents the execution time for the subtask and the number on the edge represents the communication time from the subtask to the lower subtask.
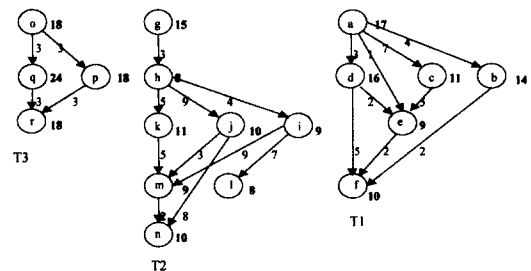
Fig 1. Periodic tasks T3, T2, T1(period 60,120,80)

Two pseudo nodes R and T are added to form an extended task graph in length of LCM in Figure 2.
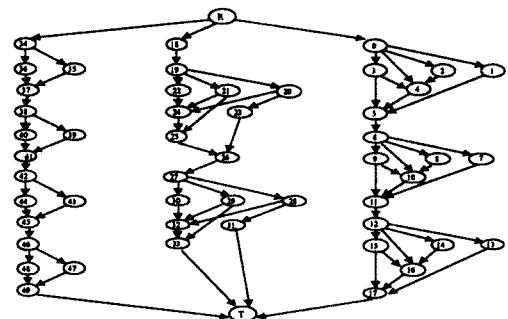
Fig 2. A Task graph extended to LCM length

To simplify the notation, we would like to define the identifier(id) of subtasks to be the arbitrary English character which is identical with the existing papers[17].

3.2 System environment

In this paper, we suppose that the system is the one which consists of the m number of functionally homogeneous processors and that lots of processors are connected via a communication network. Then the systems will be executed according to their requested task order, and the processors are not restricted in regard to the duplication scheduling algorithm[12]. The attributes of the presented graph, the precedent conditions for the execution and the time constraints are followed to the [5,7,9,10,17].

3.3 Subtask Deadline Assignment

Assumptionally, unless the subtask's deadline is defined, the deadline of the subtask is equal to that of the entire task[10]. As the first step to find out the deadline of subtasks, we will use the time constraints of the task graph that are extended to the LCM length shown in Figure 2. The $EST(\tau_{i,j})$ value of the jth instance of each τ_i is the earliest possible starting time of the subtask $\tau_{i,j}$ after the execution time $\tau_{i-1,j}$ and the communication time between $\tau_{i-1,j}$ and $\tau_{i,j}$ has passed. So, the $EST(\tau_{i,j})$ value is the sum total of the $EST(\tau_{i-1,j})$, the execution time $ex(\tau_{i-1,j})$, and the communication time $comm(\tau_{i-1,j}, \tau_{i,j})$.

$$EST(\tau_{i,j}) = EST(\tau_{i-1,j}) + ex(\tau_{i-1,j}) + comm(\tau_{i-1,j}, \tau_{i,j}) \quad (1)$$

As the result of finding out the $EST(\tau_{i,j})$ value of the subtasks applied to formula (1), the EST value means the maximum value from each of the EST values, and is computed for the subtasks with more than one predecessor.

First, the ESD(Effective Serial subtask Deadline) allocation strategy is to multiply the total processor idle time left from the current subtask to the last subtask by the processor idle time allocated to an arbitrary subtask of the entire deadline. Then add the value EST and ex(the execution time). And then we can get a effective deadline concerning the communication time and the execution time in formula (2).

$$dl(\tau_{i,j}) = \lceil EST(\tau_{i,j}) + ex(\tau_{i,j}) + \underbrace{\left[\underbrace{\left(deadline(T_i) - EST(\tau_{i,j}) - \sum_{k=i}^{m} ex(\tau_{k,j}) \right)}_{②} \times \underbrace{ex(\tau_{i,j}) / \sum_{k=i}^{m} ex(\tau_{k,j})}_{③} \right]}_{①} \rceil \quad (2)$$

① The total processor idle time assigned to the τ_i

② The left time of processor idle time

③ The rate of the execution time

In the extended graph in Figure 2, the serial subtask's deadline in the first instance of the ith subtasks of T_1 is computed as follows: dl(2)≃50, dl(3)≃55, dl(4)≃70, dl(19)≃48, dl(20)≃73, dl(21)≃75, dl(22)≃75, dl(23)≃110, dl(24)≃110.

Let's look at the another strategy called EPD (Effective Parallel subtask Deadline) allocation which can be executed in parallel at any level of the task graph. The deadline of the $\tau_{i,j}$ of the parallel subtask can be calculated as follows: first, subtract EST from the deadlines of all the tasks, and divide the value by the number of parallel executing subtasks.

$$dl(\tau_{i,j}) = \lceil [deadline(T_i) - EST(\tau_{i,j})] / \text{the number of subtask} + EST(\tau_{i,j}) \rceil \quad (3)$$

According to formula(3), the deadline of the parallel subtasks 1, 35, 36 are as follows: dl(1)≃34, dl(35)≃41, dl(36)≃41. If there is more than one deadline like join or fork subtasks, then we choose the largest of all deadlines computed. At the second instance, as $EST(\tau_{i,j}, \tau_{i,j+1})$ of the τ_6 is equal to $EST(\tau_{i,1})$ plus $p(T_i)$, the value of the $EST(j,2)$ is set to 80 and then the deadline is computed. The SDA(Subtask Deadline Assignment) algorithm in-

cluding the ESD and the EPD allocation strategies which were described previously is shown in Figure 3.

```
SDA algorithm
 (input X :task; D:deadline)
  begin
   if X is a simple subtask then            //ST_i
        if X is a root then
           dl(X) :=start_time (X) + ex(X)
        else dl(X):= D
   else if  X = [X_j, X_{j+1}, ..., X_n] then   //SST_i
            assign deadline to task X,
            according to the ESD strategy.
                 //from equation(2)//
            SDA (X_j, dl(X_j));
        else if X = [X_1 || X_2 || ...X_n] then//PST_i
                 for i = 1 to n
                 assign deadline to X,
                 according to the EPD strategy.
                 //from equation(3)//
                 SDA (X_i, dl(X_i)) ;
   endif
 end
```

Fig 3. SDA algorithm formed with ESD and EPD

3.4 Processor allocation algorithm

In this section we propose the processor allocation algorithm based on a heuristic method, using the parameters of a real-time timing characteristics that were previously computed. The principle of the allocation, first of all, is to assign the ready tasks in the sequence of the priority queue to the processor.

3.4.1 Non-duplication allocation algorithm

Figure 4 shows the allocation algorithm that considers only the time characteristics without duplication, and we suppose the starting time of all processors is set to zero, and the id of allocated subtask is set to nil.

The status information of each processor is starting time of the allocated subtasks in any processor, finish time which is the sum of start time and execution time and currently id of allocated subtasks. The status information of task allocation for processor P_i is listed in Table 1.

The result allocated to the processors with regard to the time characteristics of the subtasks is shown in Figure 5. As a result of the pre-allocation of each subtask to the processor, each processor utilization U_i is computed by equation (4) with the sum of the execution time and the rate of the LCM in the processor P_i.

$$Processor\ Utilization(U_i) = \frac{[\sum_{j=1}^{m} \phi(\tau_j)]}{LCM} \quad \text{for } P_i \quad (j=1,2,\cdots\cdots,m) \tag{4}$$

where, $\phi(\tau_j)$ is the execution time of each subtask τ_j which is assigned to a processor. The periods of task T_1, T_2, T_3 are 60, 120, 80 time unit respectively. When the system is extended, the total time unit of task periods within one scheduling frame will be 240 time unit. The detailed results for the non-duplication allocation are shown in Figure 5. For example, the U_1 for the processor P_1 case is

$$\frac{[\sum \phi(\tau_{34}, \tau_{36}, \tau_{37}, \tau_{38}, \tau_6, \tau_7, \tau_{26}, \tau_{27}, \tau_{28}, \tau_{31}, \tau_{12}, \tau_{15}, \tau_{16}, \tau_{17})]}{240}$$

$$\text{for } P_1 = \frac{201}{240} = 0.8375$$

After the tasks are scheduled, we can obtain the processor utilizations of the eight processors in Figure 5 as follows : $U_1 = 0.8375$, $U_2 = 0.8916$, $U_3 = 0.3916$, $U_4 = 0.4708$, $U_5 = 0.2958$, $U_6 = 0.23333$, $U_7 = 0.0458$, $U_8 = 0.1208$. Each of the dark areas

Table 1. The information status of subtask allocation

P_{id}	Status information														
P_1	start_time	0	18	42	60	80	97	120	135	143	152	160	177	193	202
	finish_time	18	42	60	78	97	111	135	143	152	160	177	193	202	212
	id of allocated subtasks	τ_{34}	τ_{36}	τ_{37}	τ_{38}	τ_6	τ_7	τ_{26}	τ_{27}	τ_{28}	τ_{31}	τ_{12}	τ_{15}	τ_{16}	τ_{17}
⋮	⋮														

```
Non-duplication algorithm

Process Information{

struct

        proc_id;              //id of processor
        subtask_id;           //id of subtask which is assigned to
        finish_time;          //finish time of any subtask
        start_time;           //start time of any subtask
                }
Process Information Do_subtask    //information for tasks

priority_queue_subtask[i]{
        EST(Earliest start_time of any subtasks)
        ex(execution time)
        pred(predecessor)
        comm(communication time from predecessor to sucessor)
        }
proc_start_time[]                 //start_time of each processor
 for(i=0, j=0; i<max_num; i++){
                              //in case of the predecessors are not exist
  if(subtask[i].pred==nil){
    Do_subtask[j].subtask_id=i;
    //the most short star_time in all processors in use will be
      selected assigned into min_proc_start_time, id is assigned
      into min_proc_id
    Do_subtask[j].finish_time=min_proc_start_time+subtask[i].ex;
    Do_subtask[j].proc_id = min_proc_id; }

  else{                       //in case of the predecessors are exist
    //subtask[i].pred is found in Do_subtask[] array
      available start_time and to search for pre.finish_time and pred.proc_id
      min_proc_start_time=∞
      min_proc_id=nil
      for(k=0; k<allocated processor number+1; k++)
       {
        if(k==pred.proc_id)            //communication time is zero
          start_enable_time=proc[k].start_time; }
        else{                          //there exists communication time
             if(proc_start_time>=(pred_finish_time+subtask[i].comm)
                  start_enable_time=proc[k].start_time;
             else
                  start_enable_time=pred_finish_time+subtask[i].comm;
            }
        }
      if(min_proc_start_time>start_enable_time){
           min_proc_start_time=start_enable_time;
           min_proc_id=k; }
                :
Do_subtask[i].subtask_id=subtask[i].id;
Do_subtask[i].start_time=min_proc_start_time;
Do_subtask[i].finish_time=min_proc_start_time+subtask[i].ex;
Do_subtask[i].proc_id=min_proc_id;
```

Fig 4. Non-duplication allocation algorithm

in Figure 5 indicate the IPC(Inter Processor Communication)time, and we can see where the overall tasks violate the deadline of the three time units.

3.4.2 Allocation algorithm allowing duplication

The task duplication in the existing real-time system was considered to the deadline of each
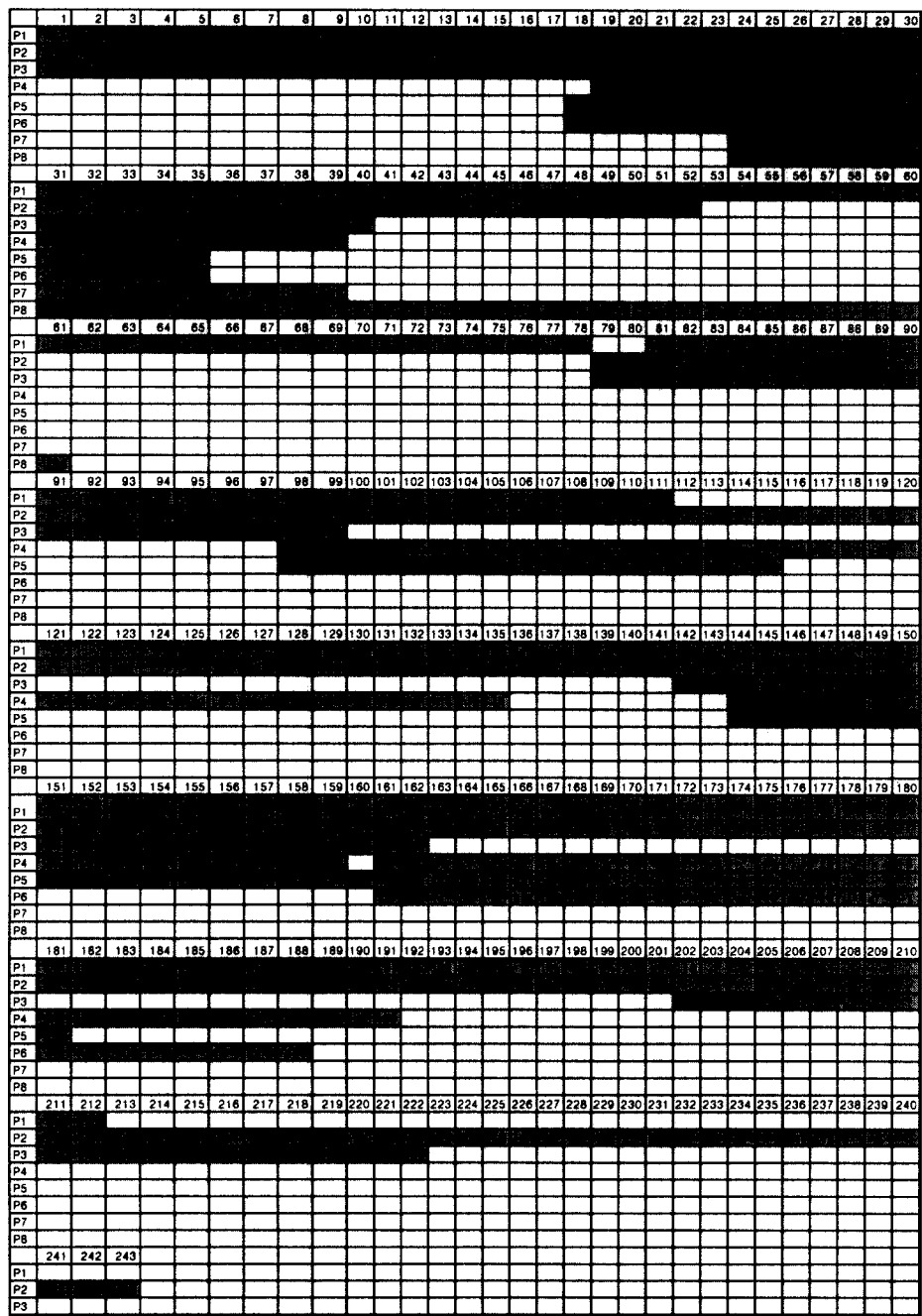
Fig 5. Allocated result applied to non-duplication allocation

tasks [9,17], but we considered the duplication to be an improvement of the processor utilization of the serial-parallel subtasks. We allocated the tasks to the processors by using the following EDFRN (Effective Duplication Frist and Reduction Next) method, and then compared it with the existing method. The computed EST value is used to decide the priority of each subtasks. To be describe the

meaning of the variables used in that algorithm, BC(Bottom-up Computation) is the amount of the communication time and the execution time from the end task to τ_i. CIP is the critical immediate parent that has the largest MAT(Message Arriving Time), which is the completion time plus communication time from each immediate parent. ECT is the earliest completion time τ_i which can carry out. IP is the immediate parent. JT is the join task which has more than one immediate parents. LT is the most recent task assigned to each processor P_i. MAT is the message arrival time from τ_{i-1} to τ_i . P_a is the processor. P_c is the critical processor that CIP is assigned to. P_u is the idle time processor without the task allocation at the point in time when the arbitrary task is scheduled. And P_n expresses the new processors respectively.

1) FUTD algorithm

In the algorithm of the FUTD(Fully connected, Unbounded Task Duplication) in Figure 6, first of all, we cluster the task by using linear clustering, and BC value is used to determine the priority of the tasks. The tasks in the cluster are inserted to the ready queue when they are ready. The sequence of inserts are done in descending order according to their computed BC value, and the processor number which is allocated to the task correspond to the cluster number. Then we start assigning the tasks to the processors, and continue to execute until the ready queue is empty.

Getting the BC value, which is the priority of each task, and assigning the subtasks to the processor by applying the FUTD to it, we can obtain the processor utilizations of the 17 processors as follows: $U_1=1$, $U_2=0.4416$, $U_3=0.588$, $U_4=0.167$, $U_5=0.167$, $U_6=0.15$, $U_7=0.225$, $U_8=0.225$, $U_9=0.225$, $U_{10}=0.138$, $U_{11}=0.138$, $U_{12}=0.129$, $U_{13}=0.13$, $U_{14}=0.171$, $U_{15}=0.179$, $U_{16}=0.171$, $U_{17}=0.179$. Concerning the completion time of executing the entire task, the execution is completed in 240 time units due to the improved the deadline miss before the duplication. 23 time units of IPC time decreases, but if immediate parents are not LT, it has the demerits that the more the task graph is deep, the more the number of the processors increases because the subtasks continue to be allocated to the new processors.

```
FUTD algorithm

if τi is not a Join task
  if  not scheduled IP onto Pa and unused Pa
      copy the scheduled up to IP onto Pa
           //Pa is corresponding the processor Ca
  endif
        schedule τi to Pa
      else  try_duplication(Pa, τi)
            schedule τi to Pa
endif
```

Fig 6. FUTD Algorithm

2) EDFRN algorithm

The EDFRN(Effective Duplication First and Reduction Next) algorithm in Figure 7 considers the new processor after considering the descending order of the idle time processor at the point of time when the subtask is assigned to the current target processor.

```
EDFRN algorithm
initialize()         //build a priority queue using EST//
for each subtask τi in the queue //in FIFO manner//
  if τi is not a Join task     // τi has only one IP//
      identify the IP
      if the IP is LT
        schedule τi to the P having the IP
      else if there exists having the Pu
             copy the schedule up to the  IP onto Pu
             schedule τi to Pu
           else copy the schedule up to the IP onto  Pn
                schedule τi to Pn
      endif
  else                              //if τi is a Join task
      identify CIP and Pc
      if CIP is LT
        DFRN(Pc , τi )      //apply DFRN to Pc
        schedule τi to the Pc having the CIP
      else                          //if CIP is not LT
        copy the schedule up to CIP onto Pn
        DFRN(Pn , τi )      //apply DFRN to Pn
        schedule τi to the Pn
      endif
  endif
endfor
```

Fig 7. EDFRN Algorithm

The EDFRN algorithm is a method of using the processor effectively through minimizing number of processors, which is the demerits of FUTD. As a result of executed by EDFRN algorithm, the deadline miss of the entire task does not happen, the interprocessor communication time reduces, and the processor utilization increases.

In Figure 8, each subtask is assigned to the processors by applying EDFRN. From the result we obtained the 9 processors utilization as follows
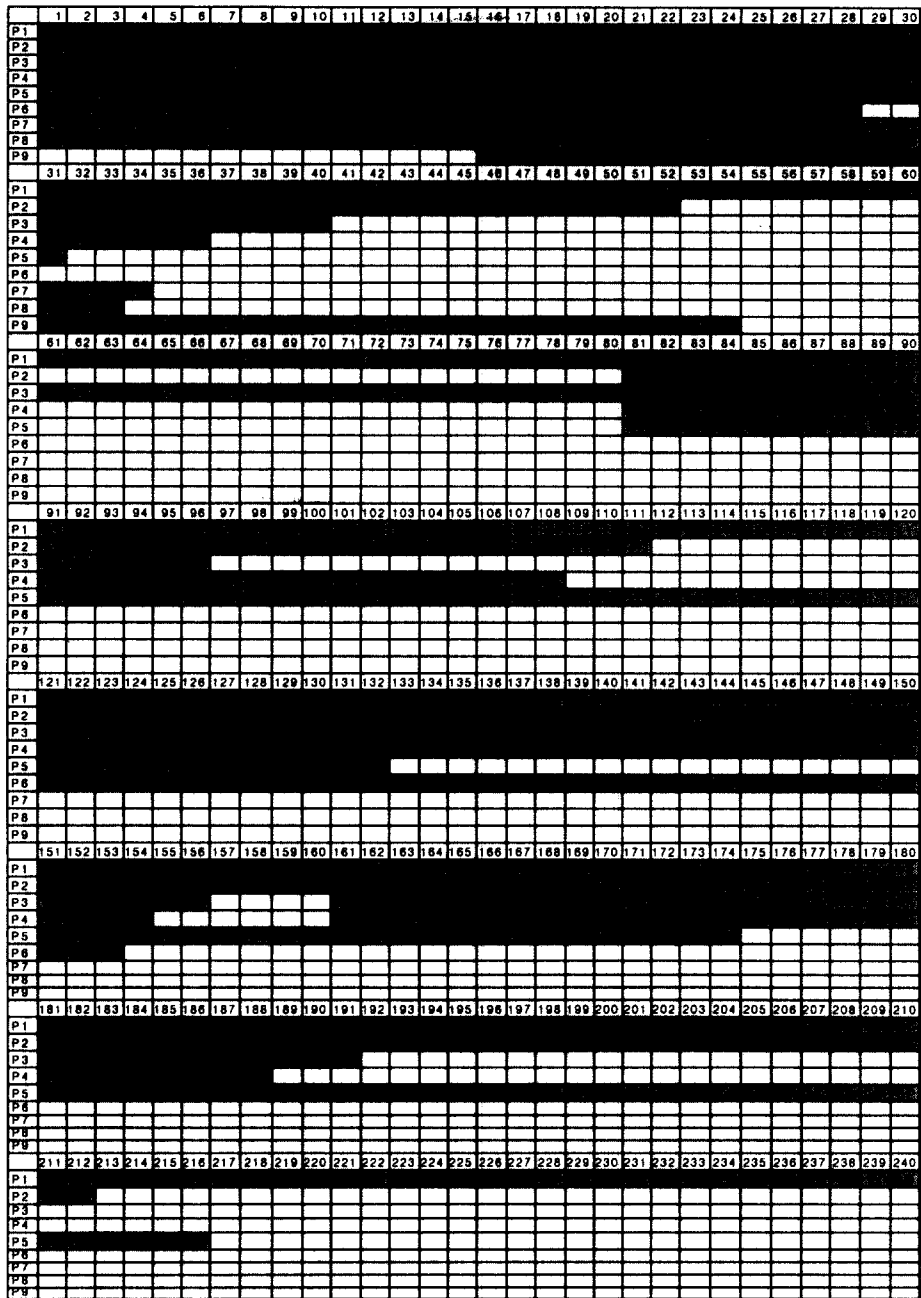
Fig 8. Allocated result applied to EDFRN algorithm

: $U_1 = 1$, $U_2 = 0.73$, $U_3 = 0.6$, $U_4 = 0.525$, $U_5 = 0.645$, $U_6 = 0.254$, $U_7 = 0.1416$, $U_8 = 0.1375$, $U_9 = 0.15$. The execution completion time of the entire task is the same as the FUTD, but the IPC time reduce to 35 time units. Finally, compared with non-duplication method, EDFRN method did not violate the deadline in the execution completion time of the entire task, and reduced the IPC time of 58 time units.

4. Performance evaluation

To measure the assignment performance of the proposed subtask deadline assignment algorithm and the duplication allocation algorithm, like the existing paper[17], we supposed that the task graph had 1~3 periodic tasks with different periods, 4~10 subtasks, and carried out the simulation so that the communication time between each subtask of the 1~10 time units and the execution time of the 1~25 time units were applied. The results of the allocations before and after duplication executions allowed considerable improvements in the processor utilization as shown in the Table 2. In case of FUTD, the average processor utilization rate before the duplication showed a decrease of 1%, but the proposed method showed an increase of 10%. Also, the executing completion time of the entire task was identical with that of FUTD, but IPC time was less than that of FUTD.

Figure 9, 10 shows that the EDFRN improved the average processor utilization and the entire task deadline miss, compared with those before and after duplication.
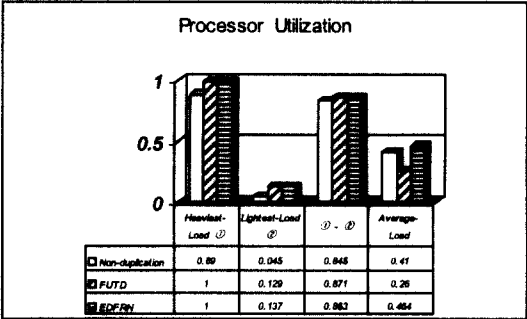
Fig 9. The processor utilization before and after duplication
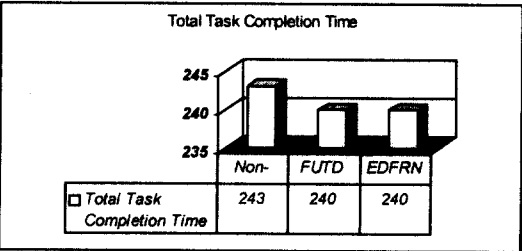
Fig 10. The Entire task executing completion time before and after duplication

Figure 11 shows that the proposed EDFRN reduces the 58 time units of the IPC time before and after duplication.

5. Conclusion

This paper focused on the problem of subtask

Table 2. Comparison allocation result before and after duplication.

		Before Duplication	After Duplication	
			FUTD	Proposed
Processor Utilization	Heaviest Load ①	0.89	1.0	1.0
	Lightest Load ②	0.045	0.129	0.137
	① - ②	0.845	0.871	0.863
	Average Load	0.41	0.26	0.464
Total Task Completion Time		243 time unit	240 time unit	240 time unit
IPC Time		64 time unit	41 time unit	6 time unit
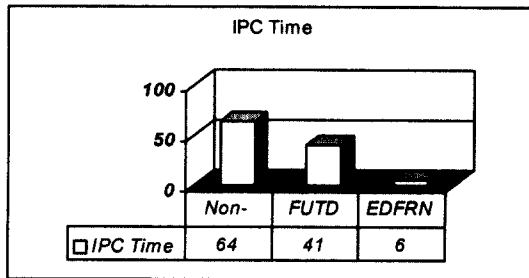
Fig 11. The IPC time before and after duplication

deadline assignment in a distributed real-time system. We transformed the task graph extended in LCM length on the basis of the EST value of the real-time timing characteristics. We proposed the effective deadline assignment algorithm considering the communication time and the execution time between subtasks according to their types, minimized the deadline miss rate of subtasks. Also, we proposed the task duplication assignment algorithm which allowed the real-time timing characteristic of the existing duplication method to obtain a faster execution completion time by minimizing the communication time between the processors.

As the result of the assignments we could, reduce the IPC time by applying the real-time timing characteristics to FUTD in which the duplication stage was constructed with the 2 stages, improve the average processor utilization and decrease the processor utilization deviation by using the idle time processors more effectively. In future research, we will focus on the development of a more comprehensive duplication method which considers the non-periodic tasks and its software tools along with the simulation environment.

References

[1] Ben Kao and Hector Garcia-Molina, "Deadline Assignment in a Distributed Soft Real-Time System", TR, Department of Computer Science, University of Stanford at stanford, April 1993.

[2] Ben Kao and Hector Garcia-Molina, "Subtask Deadline Assignmement for Complex Distributed Soft Real-Time tasks", The 14th ICDCS, Poznam, Poland, pp.172-181, June 1994.

[3] Gyung-Leen Park, Behrooz Shirazi and Jeff Marquis, "DFRN: A New Approach for Duplication Based Scheduling for Distributed Memory Multiprocessor Systems", In the Proceedings of 11th International Parallel Processing Symposium, pp.157-166, 1997.

[4] Riccardo Bettati, "End-To-End Scheduling to Meet Deadlines in Distributed Systems", PhD thesis, University of Illinois., 1994.

[5] Krithi Ramamritham, "Allocation and Scheduling of Precedence-Related Periodic Tasks", IEEE Transactions on Parallel and Distributed Systems, Vol. 6, No. 4, pp.412-420, April 1995.

[6] Mark H. Klein, John P. Lehoczky, and Ragunarithan Rajkumar, "Rate-Monotonic Analysis for Real-Time Industrial Computing", IEEE Transactions on Computer, pp.24-32, January 1994.

[7] Krithi Ramamritham, "Allocation and Scheduling of Complex Periodic Tasks", IEEE 10th International Conference on Distributed Computing Systems, pp.108-115, 1990.

[8] Rhan Ha and Jane W.S. Liu, "Validating Timing Constraints in Multiprocessor and Distributed Real-Time Systems", In the Proceedings of IEEE 14th International Conference on Distributed Computing Systems, 1994.

[9] Sheng-Tzong Cheng and Ashok K. Agrawala, "Allocation and Scheduling of Real-Time Periodic Tasks with Relative Timing Constraints", TR, Department of Computer Science, University of Maryland at College Park, January 1995.

[10] Stefan Rönngren and Behrooz A. Shirazi, "Static Multiprocessor Scheduling of Periodic Real-Time Tasks with Precedence Constraints and Communication Costs", Proceedings of the 28th Annual Hawaii International Conference on System Sciences, pp.143-152, 1995.

[11] John A. Stankovic, "Real-Time Computng", TR, Department of Computer Science, University of Massachusetts at Amherst, April 1992.

[12] Jose Javier Gutierrez Garcia and Michael Gonzalez Harbour, "Optimized Priority Assignment for Tasks

and Messages in Distributed hard real-time systems", IEEE pp.124-132, 1995.

[13] Chao-Ju Hou and Kang G. Shin, "Replication and Allocation of Task Modules in Distributed Real-Time Systems," IEEE 24th Annual Int'l Symposium on Fault-tolerant Computing, pp.26-35, Austin, Texas, June 15-17, 1994. An enhanced version has seen submitted to IEEE Trans. on Parallel and Distributed Systems as a technical brief, July 1995.

[14] I. Ahmad, Yu-Kwong Kwok, "A new approach to scheduling parallel programs using task duplication," Int'l Conf. on Parallel Processing, pp.Ⅱ47-Ⅱ51, 1994.

[15] S. Darbha and D.P.Agrawal, "SDBS: A task duplication based optimal scheduling algorithm," Proc. of Scalable High Performance Computing Conf., pp.756-763, May 1994.

[16] Joo-Man Kim, Chee-Hang Park and Cheol-Hoon Lee, "An Efficient Task Assignment Algorithm in Distributed System," The Transactions of the Korea Information Processing Society, Vol. 5, No 2, Feb. 1998.

[17] Sheng-Tzong Cheng, Shyh-In Hwang and Ashok K. Agrawala, "Mission-Oriented Replication of Periodic Tasks in Real-Time Distributed Systems," Technical Report CS-TR-3256, Department of Computer Science, University of Maryland, College Park, 1994.

[18] Kyung-Hoon Jeong, "A Study on the Task Duplication Scheduling Algorithms in Distributed and Parallel Systems", M.S., thesis, Department of Computer Science, National University of Pukyong, Feb. 1998.

[19] Min-Hwan Jo and Jeong-Yeon Huh, "Optimal Task Duplication Scheduling with a Variable Communication Time," The Korea Information Processing Society, Spring, 1998.

박 미 경

1996년 한국방송대학교 전자계산학과(이학사)
1998년 부경대학교 전산정보학과(이학석사)
1999년~현재 부경대학교 전자계산학과 박사과정
관심분야 : 분산병렬처리, 실시간 시스템, 라우팅 등

김 창 수

1984년 울산공과대학 전산학과 졸업(공학사)
1986년 중앙대학교 전산학과 졸업(이학석사)
1991년 중앙대학교 전산학과 졸업(공학박사)
1992~1996. 7 부산 수산대학교 전산학과 전강·조교수
1994~1996 국립수산진흥원 겸임연구관
1997년 미국 UCSB 전산학과 방문교수
1996~현재 부경대학교 컴퓨터멀티미디어공학부 부교수
관심분야 : 분산병렬처리, 통신보안, GPS/GIS 응용, 인터넷 DB 등